This is an accepted version of the following published document: G. Juez, E. Amparan, R. Lattarulo, J. P. Rastelli, A. Ruiz and H. Espinoza, "Safety assessment of automated vehicle functions by simulation-based fault injection," 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Vienna, 2017, pp. 214-219., https://doi.org/10.1109/icves.2017.7991928

Safety Assessment of Automated Vehicle Functions by Simulation-based Fault Injection

Garazi Juez[1], Estibaliz Amparan[1], Ray Lattarulo[1], Joshue Perez Rastelli[1], Alejandra Ruiz[1] and Huascar Espinoza[1]

***Abstract*— As automated driving vehicles become more sophisticated and pervasive, it is increasingly important to assure its safety even in the presence of faults. This paper presents a simulation-based fault injection approach (Sabotage) aimed at assessing the safety of automated vehicle functions. In particular, we focus on a case study to forecast fault effects during the model-based design of a lateral control function. The goal is to determine the acceptable fault detection interval for permanent faults based on the maximum lateral error and steering saturation. In this work, we performed fault injection simulations to derive the most appropriate safety goals, safety requirements, and fault handling strategies at an early concept phase of an ISO 26262-compliant safety assessment process.**

I. Introduction

New technologies like driverless vehicles have the potential to bring major benefits to road transportation. Due to the fact that the world is hazardous and complex, there is no hesitation that building such robust and dependable highly automated systems leads to many technological challenges as never known before. Those challenges are constituted as three main pillars underpinning the way towards dependable systems of systems: (i) novel technologies and architecture solutions are needed to reach a certain degree of dependability [1], (ii) dependability assurance is getting more complicated and (iii) novel techniques are required to help assuring dependability concerns.

As stated, the complexity of safety assessments increases with the number of non-predictable situations. Up to today, that process has been mainly driven by analyses techniques such as FMEA (Failure Modes and Effect Analysis), FTA (Fault Tree Analysis) or DFA (Dependent Failure Analysis). However, sometimes a failure effect is not clearly known in advance. Furthermore, to achieve the highest integrity levels not only safety analyses but test-based evidences proving those statements are required. In brief, to cope with that complexity, the aforementioned techniques shall be completed with Fault Injection (FI).

So far, few studies have focused on the intensive testing of autonomous systems in realistic simulation environments [2]. Building up a robust simulation framework, where vehicle dynamics is combined with FI, would allow an early safety assessment supported by safety and controllability related simulation data. According to Koopman [3], controllability stands for ensuring the ability to avoid a specified harm or damage through the timely reactions of the vehicle, assuming the driver is out of the loop. In contrast, ISO 26262 [4] introduced a different controllability definition, centred on the driver's ability to control the vehicle. The next version of ISO 26262 will likely need to adjust the controllability definition to highly automated driving (HAD).

A promising approach is to use FI [5] [6] already in early development phases. Considering engineering needs to prove fault reaction time being shorter than FTTI (Fault Tolerant Time Interval) and any fault reaction is completed before a hazardous event occurs, a good estimation of those values is relevant. In fact, academia, industry and ISO 26262 have mostly considered FI as a way to verify how good safety mechanisms are designed (Fault Removal) but not for dependability evaluation (Fault Forecasting) already at ISO 26262-3 concept phase which is the main goal of our work. For instance, Svenningson [7] investigated the benefits from conducting FI experiments on behavioural models of software. This approach is defined as model-implemented FI, since a model is extended with artefacts to support the injection of fault effects during simulation. Another similar approach is introduced in [8] which is developed as a plug-in to SCADE (Safety-Critical Application Development Environment). Although those approaches aim at determining failure effects, they cannot be acquired at vehicle level since no vehicle dynamics is taken into account. Moreover, no controllability values are estimated and no concept phase issues are tackled. The most similar work to ours, it is addressed by Silveira et al. [9] who described a co-simulation framework including Matlab and CarSim (vehicle dynamics) for evaluating the stability of electrical vehicles using FI. However, this latter work did not analyse controllability challenges and fail-operational issues, which are relevant factors for automated driving.

Only few works tackled FI-usage during early design phases. Pintard [10] established how FI can be applied on both sides of ISO 26262's V-Cycle. Yet its usage at concept phase, the development of a simulation-based FI framework and controllability issues are out of scope.

The work underlying this paper intends to present our simulation-based FI framework called Sabotage applied at the ISO 26262 concept phase. This includes forecasting fault effects, getting representative FTTIs and deriving the safety goals and the functional safety requirements. Our approach has been evaluated on a Lateral Control.

The rest of this work is organized as follows: Section II

[1]ICT and Automotive Divisions, Tecnalia Research and Innovation, Derio, Vizcaya, Spain, firstname.lastname@tecnalia.com

presents the automated vehicle control architecture. Thereafter, in Section III we describe the Sabotage tool framework and how to apply it during the concept phase of ISO 26262. Then, the previous methodology is applied to a Lateral Control. Finally, Section V presents conclusions leading to an outlook on future work.

II. Automated Vehicle Control Architecture

To better understand the current situation concerning highly automated vehicle (HAV) architectures, an overall overview of them will be introduced. Nowadays, the algorithms embedded on those applications are marked by the integration of different subsystems on a modular architecture. This separation in different modules reduces the time for troubleshooting possible failures but at the cost of more complicated allocation of functions. As a consequence, properties such as *freedom from interference* [4] turns into a more complicated issue to prove. The architecture is mainly divided into the *Lateral* and *Longitudinal Control*. The aim of the *Lateral Control* is to steer the vehicle along the best trajectory of the circuit. For this purpose, the steering wheel is adjusted by choosing the correct trajectory depending on the vehicle and environment states. This item is composed of three essential functions:

- Behavioural Planner chooses the best trajectory depending on the vehicle manoeuvre (i.e. lane keeping, lane changing or avoiding an obstacle). It consists of three main sub-functions: *Local Planner*, *Perception* and *Decision*. The *Local Planner* gathers information from sensors so the current position value on the road is computed by the adequate processing. The *Perception* sub-function gets data from the vehicle (e.g. encoders, IMU) and environment state sensor (e.g. Camera and Lidar) fusing the information of the environment around the vehicle. At last, the *Decision* sub-function generates the best trajectory tracked by the vehicle.
- Trajectory Controller tries to keep the vehicle on the correct trajectory. To do so, the lateral error, the angular error and the curvature of the path are sent to the steering function. The algorithm chosen for the evaluation of this design is the so-called Control Law algorithm and it is defined by the following (1):

$$C_v = K_1 * e_{lat} + K_2 * e_{ang} + K_3 * Curvature \quad (1)$$

- Steering Function controls the steering wheel to place the vehicle on its trajectory. It receives the corrected value as input from the Trajectory Controller.

Whereas the Lateral Control is responsible for steering, the *Longitudinal Control* maintains the vehicle along its trajectory by managing the acceleration and braking systems. It is divided into Behavioural Planner, Trajectory Controller, Acceleration and Braking functions (out of scope).

III. Sabotage Framework According to ISO 26262

A. Sabotage: block diagram

Once the architecture for HAD has been introduced, it is important to understand why FI can be seen as a complementary approach when evaluating the safety of HAV functions. As previously stated, FI establishes itself as a promising technique to evaluate the safety and controllability of such systems. In fact, compared to formal methods or mutation testing (removal of systematic faults), our work evaluates the effect and fault handling of faults occurring during run-time. Thus, Fig. 1 illustrates our approach based on a simulation-based FI solution for safety assessment of automated vehicle functions. Since not all controllability values and failure effects can be known in advance, our approach provides assistance (i.e. fault forecasting) to evaluate the safety of a certain architecture in early design phases. By using that simulation data, a trade-off between the most suitable safety concepts is possible. Even though FI sounds as a promising approach, it requires the user to have a basic knowledge of the system i.e. random FI might not be feasible in practice.

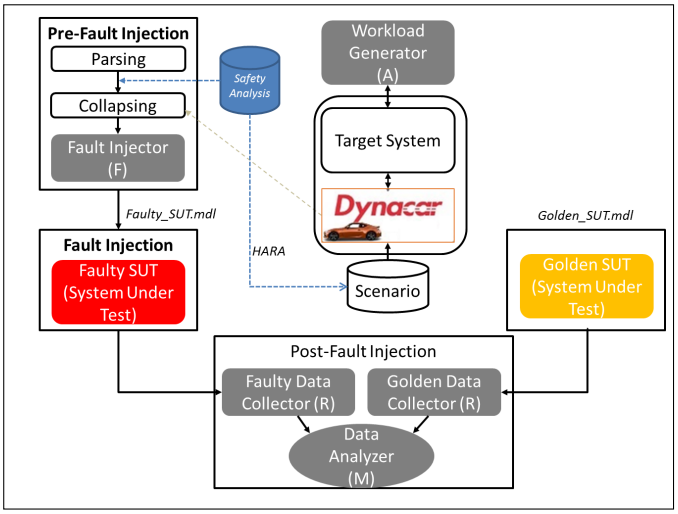

Fig. 1: Sabotage: Simulation-based FI Framework

After pointing out the main benefits of performing FI, the main flow of our approach will be explained. To begin with, the main functions of the item must be established and its malfunctions (e.g. omission) specified. Then, the functional failure modes are properly identified in order to get data regarding its effects (at system/vehicle level). This means that if those failure modes are defined at system level its effects are seen at vehicle level. On the contrary, if the functional failure modes are specified at component level, those effects would lead to system and vehicle level failures/hazards. Those malfunctions/failure modes are related with a generic fault model saved on the so-called generic fault model library (Omission, Frozen, Delay, Invert, Oscillation, Random). These generic fault models are prearranged and their role is to simulate specific fault models of any component/system functional failure mode. The creation of these templates helps the user to perform the relation between any failure mode and its respective generic fault model template, ensuring the user does not provoke systematic faults. For example, the code of a generic Omission fault model is shown in algorithm 1.

Algorithm 1 Stuck at last value

Require: input,pos,simutime

```
if pos == triggerpos then
    freeze=input;
    while simutime <= triggerpos do return freeze;
else
return input;
```

After conducting a preliminary analysis of the system, the FI experiments have to be configured as part of a *Workload Generator* which includes setting the experiments and driving scenarios, and generating the *fault list*:

- *Target*: where should the fault be injected?
- *Fault Model*: what is the most appropriate fault model representing the functional failure mode?
- *Target*: where should the fault be injected?
- *Trigger*: how should the fault be triggered in the system?
- What is the fault effect observation point?
- How should the condition making the vehicle losing its controllability be defined?

Certainly, those issues are arduous to overcome and that is why the art of applying FI is neither spread enough nor mature. For each fault the user wants to inject, the information referring to *Fault Model*,*Target Signal* (Fault Location), *Fault Trigger* based on time or driving circuit position coordinates (X, Y) and *Fault Duration* must be collected within a fault list. That information is the starting point to generate the saboteur. Saboteurs are components added to system behavioural models for the sole purpose of FI. One of them is injected per target signal and its configuration regarding the fault model and the triggering condition is the one established as part of the fault list information. It is worth stressing that by injecting an unique saboteur on a target signal, different fault models might be reproduced just by changing the injection trigger.

The configuration of the experiments requires a vehicle selection and a definition of an operational situation:

- Location: highway, urban;
- Road conditions: uphill, on a curve;
- Environment conditions: good conditions, heavy rain;
- Traffic situations: fluent;
- Vehicle Speed (kph)
- Manoeuvres: parking, overtaking, lane keeping;
- People at Risk: Driver, passenger, pedestrians;

The *Scenario Configurator* chooses the best driving scenario saved in the *Scenario Catalogue*, which represents the previously defined operational situation, in order to load it into the Dynacar platform. The Dynacar platform [11] is a real-time vehicle dynamics simulation. This virtual solution based on multiple domain vehicle models provides high-fidelity of vehicle physics permitting a real time simulation combined with a notable modality and interfering options. It allows the mixing of virtual or real electronic control units, vehicle sensor and vehicle control variables. The critical systems are limited by dynamics since actuators and sensors need an interval time to respond. Through this, the FTTI values concerning vehicle level effects can be estimated.

At last, the so-called *faulty system under test* must be created. To achieve that goal, the *Fault Injector* module creates the saboteurs' code based on the information coming from the fault list together with the generic fault model code templates. This process based on libraries and lists allows to automate FI simulations. After the faulty SUT is created, its results are compared with respect to a golden (fault free) simulation run so the proper safety requirement conditions can be derived. To sum up, our approach helps the designer to avoid user faults, to get more reliable results and to save time and costs through its comprehensive automation.

B. Using Sabotage at ISO 26262 concept phase

During the set up and the design of the previously mentioned methodology, ISO 26262 has always been present. Nevertheless, the following question arises: how relevant is FI for ISO 26262? As pointed out, the current version of ISO 26262 is mostly driven by safety analyses techniques when enumerating known failure modes, deriving potential effects and getting the proper safety requirements. The same applies to the right hand of the V-model where safety analysis is used to assess the safety of the system, software or hardware.

Yet the uncertainty related to HAD makes safety analysis definitely not sufficient, requiring additional virtual and simulation solutions. As a result, FI poses as a viable and complementary solution to the aforementioned safety assessment technique. One of the main reasons behind this is the difficulty of figuring out the system reaction under the effect of real faults. The same reason applies to the derivation of the safety goals and the functional safety requirements (FSR). In few words, this methodology supports Hazard Analysis and Risk Assessment (HARA) by FI simulations. Before completing any simulation, the item should be defined and its main functions and failure modes (e.g. omission) listed. In the same way, a preliminary architecture is mandatory in order to know which functional failure modes could lead to system or vehicle failures and hazards. To set up the FI scenarios the fault list and the driving circuit settings (operating scenarios) are needed. As already depicted, after configuring the circuit settings, the aforementioned fault list is created. Once FI simulations are configured and the faulty SUT created, golden and faulty simulations are run and results processed. As a result of those simulations, the following outputs are obtained:

1) Hazard identification by FI instead of using Preliminary Hazard Analysis or FMEA. Those hazards (e.g. vehicle does not try to turn when it should) can be visually seen through Dynacar virtual environment.
2) Safety goal refinement based on simulation results and hazard identification.
3) Definition of FTTI and establishment of the safe state. As seen in Fig. 2, FTTI is the measured time from a fault is injected until a hazard can occur. Setting different fault durations, the fault detection time at system/component level and FTTI are obtained. As

the application domain is HAD, FTTI determines the required level of fault tolerance (i.e. redundancy, graceful degradation) not to lose the controllability of the vehicle.

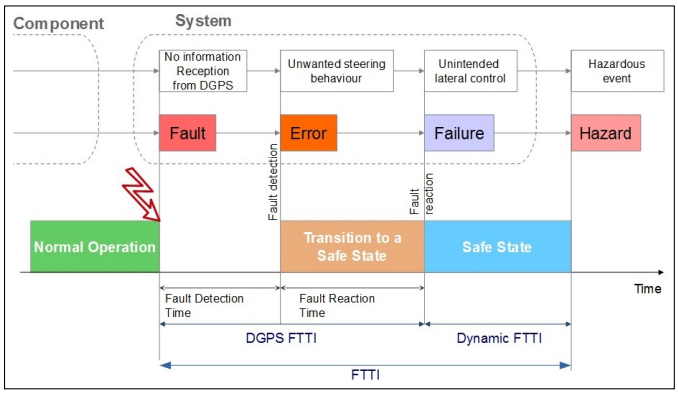

Fig. 2: Fault-Error-Failure Chain and FTTI Definition

4) Comparing golden vs faulty simulations the safety requirements can be appropriately derived as the maximum difference between the two simulations without losing the controllability of the vehicle.
5) As a consequence of the previous results, those requirements will be allocated in a functional safety concept.

IV. Safety Evaluation of the Lateral Control

As explained in Section III, automated guidance must be simultaneously performed with the *Longitudinal* and *Lateral Control*. Since ensuring functional safety is a vital requirement, this section presents a safety evaluation for an existing Lateral Control system behavioural model for which no safety mechanisms were in place. The FI simulations have been performed to address the following issues:

1) Acquire data regarding vehicle and item level effects based on FI simulation results.
2) Complete the safety analysis, define the FTTI values, the safe states, the FSRs and the safety concept. This means developing a fault tolerant Lateral Control to avoid possible hazards and ensuring a certain degree of dependability by fail-operational or graceful degradation.

A. Item Definition

As explained in Section II, the Lateral Control item can be decomposed in several functions and sub-functions and its malfunctions consist of: Steering (Ommission, Commission), Trajectory Controller (Omission, Commission), Behavioural Planner (Unwanted Local Planner, Unwanted Perception, Unwanted Decision). This information is absolutely necessary in order to properly configure the required fault models starting from the defined malfunctions.

B. Hazard Analysis and Risk Assessment

After accomplishing the item definition, HARA should be performed by evaluating it without internal safety mechanisms. As said, instead of performing it analytically, FI testing has been used as a way of determining unknown effects and deriving the safety requirements. These testing experiments require the definition of an operational situation where a specific vehicle and driving circuit scenario shall be chosen. The last one is configured considering the most critical situation for the *Lateral Control* function which is recognized as an urban city with fluent traffic. The vehicle selected for those experiments is the so-called Twizzy Renault which is driving at a constant speed of 45km/h and performing lane keeping manoeuvre. The malfunctions are triggered while the vehicle is driving on a curve. In order to see system and vehicle level effects, functional failure modes related to the DGPS (Differential GPS) and Steering system have been reproduced. The fault list (cf. Table I) is specified as follows: fault durations are randomly filled in as multiple of the simulator resolution (1ms) and triggers are curve positions (X,Y).

Component	Target	Fault Model	Fault dur	Trigger
DGPS	X,Y	FrozenLastValue	100ms	15m,20m
DGPS	X,Y	Delay	120ms	18m,24m
Steering	Steering	FrozenLastValue	80ms	15m,20m

TABLE I: Example of a fault list

By applying the process explained in Section III, the configurable saboteurs are automatically injected based on a previously built up fault list. In this case, those faults are triggered at several curve points so the most critical ones can be derived. Since our FI campaigns pursue the calculation of the FTTI values applied to the Lateral Control, the observation signals can be the lateral error and the steering saturation. For this, Fig. 3 depicts the basis for FTTI calculation in the case of the steering function of the Lateral Control.

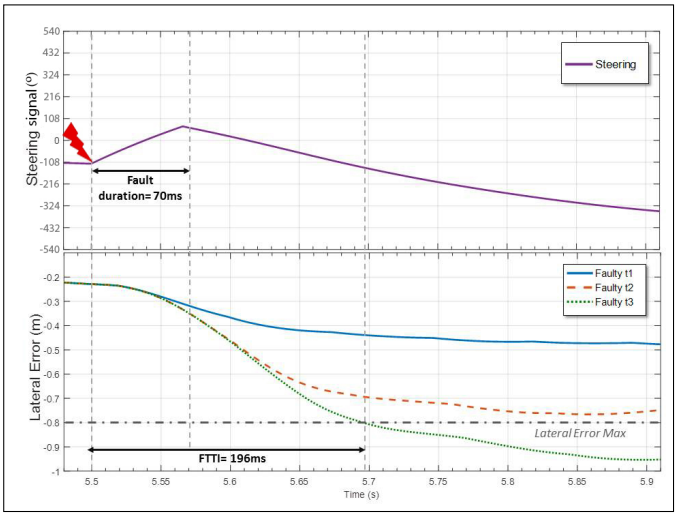

Fig. 3: Basis for FTTI calculation

To establish the controllability loss criteria, the formula 2 is used which defines the maximum lateral error:

$$LateralErrorMax = \frac{(Lane - Vehicle)_{Width}}{2} \quad (2)$$

Table II depicts how the information regarding Hazard Analysis has been obtained based on FI simulation results. To do so, item level malfunctions have been modelled by means of generic fault models which allow to measure vehicle level effects and to properly define the safety goals.

	Fault Model	Potential Effect	Hazard
Steering	FrozenOut-OfRange	Steering shaft broken	Sudden steering leading to a possible accident
	Frozen SteeringValue-Max/Min and Invert	High change of the steering angle value	Vehicle can roll/spin and passengers can be injured
	FrozenLast-Value	Frozen Steering	Vehicle out of lane
	Delay	Late Steering	Vehicle is never placed on the correct trajectory
Trajectory Controller	FrozenLast-Value	Variation Correction (Cv) does not change	Vehicle out of lane
	FrozenOut-ofRange	Controller Saturation	Vehicle is never placed on the correct trajectory
Behavioural Planner	FrozenLast-Value	The trajectory is not updated	Vehicle follows a not updated trajectory
	FrozenRandom	Behavioural Planner changes the trajectory and sudden steeering	Vehicle may spin/roll and out of the lane and passengers can be injured

TABLE II: Failure effects at vehicle level

After conducting the FI experiments, Table III has been filled in and the safety goals better elicited (cf. Table IV). Each line represents the FTTI value for the most severe failure mode (represented as a fault model) of a specific function computed as illustrated in Fig. 2 and Fig. 3. Its associated fault duration depicts the time to handle the fault in an appropriate way (transition to a safe state). For example, a fault related to the trajectory controller can be present in the system for 400 ms before a hazardous event occurs: 240 ms to detect and react, and 160 ms added time to control the fault not to violate the safety goal.

Function	Fault Model	SG	Safe State	FTTI (ms)	Fault dur(ms)
Steering	Frozen Value Max/Min	SG1	Fail-operational	196	70
Behavioural Planner	Frozen Random	SG2	Graceful degradation and control to the driver	327	155
Trajectory Controller	Frozen Out-ofRange	SG3	Graceful degradation and control to the driver	400	240

TABLE III: Lateral Control: HARA

Safety Goal	Definition
SG1	The item must never apply a high torque value related to a sudden steering
SG2	For a sensor data that supports the fact that the vehicle is lane keeping on an urban curve, the item shall establish a new trajectory without performing a sudden steering
SG3	The vehicle shall be able to properly adjust its trajectory according to a non-saturated controller value

TABLE IV: Lateral Control: definition of the safety goals

On the basis of the safety goals, FSRs have been redefined based on the FI simulation results (see Table V). The maximum lateral error is analytically calculated as depicted in (3).

$$LateralErrorMax = \frac{3,5 - 1,9}{2} = 0,8[m] \qquad (3)$$

FSR	Definition
FSR1	The LateralError must be less than LateralErrorMax
FSR2	For a velocity input less than 45 km/h and sensor data that supports the fact that the vehicle is travelling on a Urban curve, the Yaw rate shall not increase more than 15% respect to a good behaviour.
FSR3	Cv value of the controller shall not be saturated [-1, 1].
FSR4	The range of the steering shall be between [-540, 540] degrees.

TABLE V: Lateral Control: definition of safety requirements

Together with the previous results and instead of carrying out DFA by traditional techniques, simulation results have been used. The main outcome is that *freedom from interference* is not assured in the current design of the *Lateral Control*. Hence, its architecture will need to be redesigned to ensure that property. Fig. 4 illustrates those interdependencies and how some failure modes such as Unwanted Behavioural Planner and Unwanted Trajectory Controller are considered dependent failures.

C. Functional Safety Concept at Architectural Level

In the evaluation of safety concepts, one of the main outcomes is that the steering system shall be redundant in order to achieve the required level of availability. A fault related to the steering function must be controlled within 196 ms in order to prevent a hazard to happen, i.e. vehicle can roll/spin and passenger can be injured. Accordingly, the steering function must be available within 70 ms. Regarding failures related to behavioural planner, e.g. leading from DGPS faults, graceful degradation might be sufficient since fault reaction time is 155 ms. To finish with, the different functions must be properly separated to avoid cascading failures to occur.

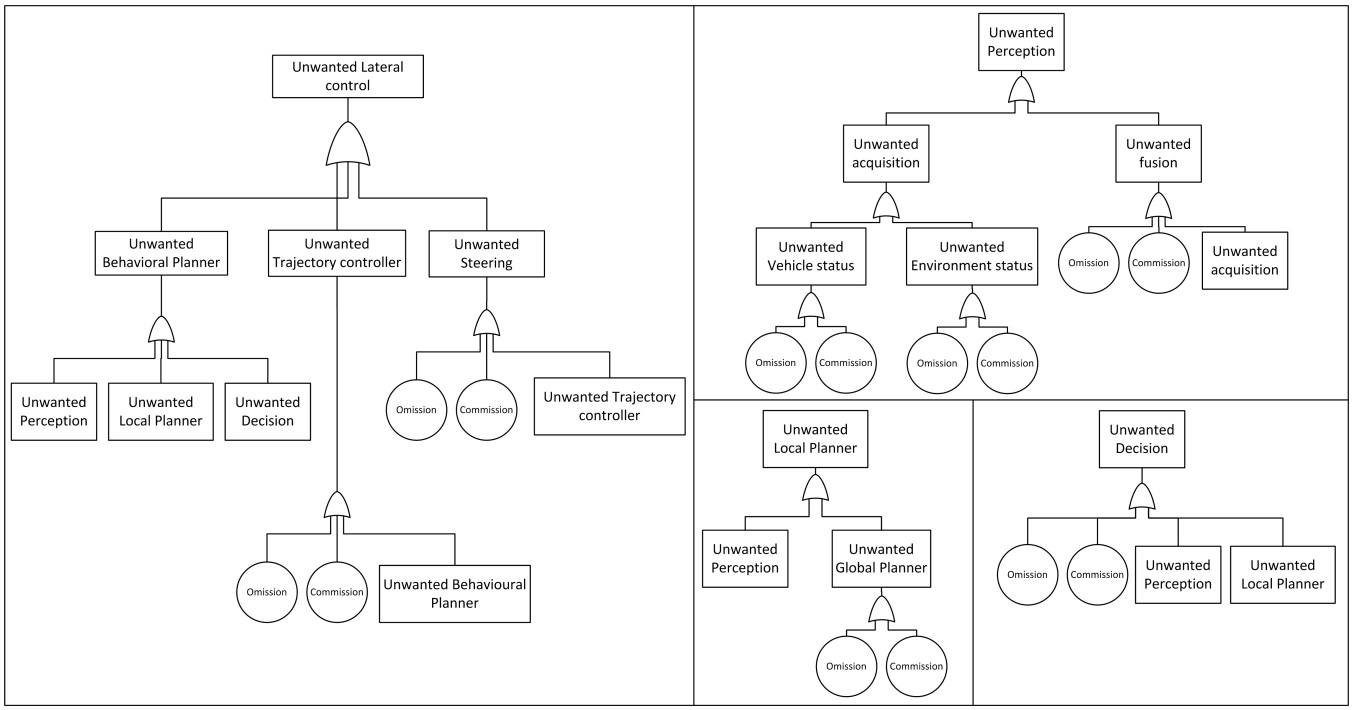

Fig. 4: Lateral Control: Fault Tree Analysis completed by FI simulations

V. Conclusion and Future Work

This paper describes the Sabotage simulation-based fault injection approach to assess the safety properties of automated vehicle functions. Simulation-based fault injection is a technique that uses a series of high-level abstractions or models representing the system under study to evaluate its dependability during conceptual and design phases. Our approach has been evaluated on a case study for the model-based design of a lateral control function embedded in an urban vehicle. From a novelty standpoint, we focused on the determination of the fault detection interval for permanent faults based on the maximum lateral error and steering saturation, as a vehicle controllability property. A major strength of the method introduced in this paper is its integration with HARA activities, which enables a seamless ISO 26262-compliant safety assessment process. Our future work aims at relaxing the fault simulation constraints to instrument the tooling automated assessment work. This includes: (1) to add the capability of collapsing and automating the injection of faults at post-processing stage, (2) the definition of generic fault models to be ready available in a database, (3) the evaluation of the acceptable time for switching the control to the driver while ensuring controllability, (4) to increase the automation of the full fault injection process from HARA to the generation of assessment reports and (5) to compare simulation results with runtime behaviour of real vehicles.

Acknowledgment

The authors have partially received funding from the ECSEL JU AMASS project under H2020 grant agreement No 692474 and from MINETUR (Spain).

References

[1] A. Ruiz, G. Juez, P. Schleiss, and G. Weiss, "A safe generic adaptation mechanism for smart cars," in *Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on*, pp. 161–171, Nov 2015.

[2] J. Guiochet, "Trusting robots: Contributions to dependable autonomous collaborative robotic systems," *Universite de Toulouse*, 2016.

[3] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *2016 SAE World Congress*, 2016.

[4] "Iso/dis 26262: Road vehicles - functional safety," 2011.

[5] A. Benso and P. Prinetto, *Fault injection techniques and tools for embedded systems reliability evaluation*. Frontiers in electronic testing, Boston, Dordrecht, London: Kluwer academic publ. cop., 2003.

[6] A. Benso and D. C. S., "The art of fault injection," *Journal of Control Engineering and Applied Informatics*, pp. 9–18, 2011.

[7] R. Svenningsson, "Model-implemented fault injection for robustness assessment," 2011.

[8] J. Vinter, L. Bromander, P. Raistrick, and H. Edler, "Fiscade - a fault injection tool for scade models," in *Automotive Electronics, 2007 3rd Institution of Engineering and Technology Conference on*, pp. 1–9, 2007.

[9] A. Silveira, R. Araujo, and R. De Castro, "Fieev: A Co-Simulation Framework for Fault Injection in Electrical Vehicles," in *2012 Ieee International Conference on Vehicular Electronics and Safety, Icves 2012*, pp. 357–362, 2012. Citations: crossref, scopus.

[10] M. L. Pintard, "Des analyses de securite a la validation experimentale par injection de fautes - le cas des systemes embarques automobiles," *PhD, Institut National Polytechnique de Toulouse*, 2015.

[11] A. Pena, I. Iglesias, J. Valera, and A. Martin, "Development and validation of dynacar rt software, a new integrated solution for design of electric and hybrid vehicles," *EVS26 Los Angeles, California*, 2012.